PL-TR-94-2012

# STOCHASTIC INDEXING

Albert R. Boehm

Hughes STX Corporation
109 Massachusetts Avenue
Lexington, MA 02173



21 January, 1994

Scientific Report No. 1



**APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.**





PHILLIPS LABORATORY
Directorate of Geophysics
AIR FORCE MATERIEL COMMAND
HANSCOM AIR FORCE BASE, MA 01731-3010

"This technical report has been reviewed and is approved for publication"

DONALD D. GRANTHAM
Contract Manager
Chief, Atmospheric Structure Branch

ROBERT A. McCLATCHEY
Director, Atmospheric Sciences Division

This report has been reviewed by the ESC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS).

Qualified requestors may obtain additional copies from the Defense Technical Information Center. All others should apply to the National Technical Information Service.

If your address has changed, or if you wish to be removed from the mailing list, or if the addressee is no longer employed by your organization, please notify PL/TSI, Hanscom AFB, MA 01731-3010. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | 21 January 1994 | Scientific Report # 1 |

**4. TITLE AND SUBTITLE**

STOCHASTIC INDEXING

**5. FUNDING NUMBERS**

F19628-93-C-0051
PE62101F
PR6670
TA GS
WU AC

**6. AUTHOR(S)**

Albert R. Boehm

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Hughes STX Corporation
109 Massachusetts Avenue
Lexington, MA 02173

**8. PERFORMING ORGANIZATION REPORT NUMBER**

Hughes STX Scientific Report #1

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Phillips Laboratory
29 Randolph Road
Hanscom AFB, MA 01731-3010

Contract Technical Manager: Donald D. Grantham/GPAA

**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

PL-TR-94-2012

**11. SUPPLEMENTARY NOTES**

**12a. DISTRIBUTION / AVAILABILITY STATEMENT**

Approved for public release: distribution unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*

A documentation of the computer program for Stochastic Imdexing (SED) is presented. A new method of "storing" and retrieving randomly generated features is described.

**14. SUBJECT TERMS**

Stochastic, clouds, storage, climatology, random numbers, random features, simulation

**15. NUMBER OF PAGES**

12

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | Unlimited |

## CONTENTS

## Foreword

This documentation is from the information file of a computer program, SID, that illustrates the concept of stochastic indexing. This information file is not intended to be a stand-alone report, but to be used with SID to quickly understand a concept that is simple for a programmer to implement but profound to a statistician in its application.

## Purpose

The demo program, SID (Stochastic Indexing Demo), in listing 1 simulates clouds as seen by a satellite looking straight down. As a pictorial rendering of clouds, it is rather poor since it only uses circles to indicate where clouds are. But with a little imagination, one can see structures that resemble some types of clouds. By varying the random number seed, the number of clouds, puffs, rows, etc., a wide variety of patterns can be generated.

However, cloud depiction is not its purpose. Its purpose is to demonstrate a new method of "storing" and retrieving randomly generated features. One hundred cloud clusters each with up to 30 puffs (an average of 17) are generated for a total of about 1700 puffs. The location and size of each of these puffs can be retrieved very fast. The computer memory storage required to "store" this information is ZERO!!

Furthermore, the number of puffs could have just as easily been 10 million with additional information on liquid water content, vertical velocity, etc. etc., and the required storage still would be zero. The method is not limited to clouds. The same method could be used for simulated trees in a forest, simulated fish in the sea, or simulated accounts in a bank.

The trick is to reseed the random number generator using the feature's index - in this case the cloud's identification number, 1 to 100. Some background on random number generators may help understanding.

## Random Number Generators

Computer random number generators start with some initial number, operate on this number, then return the result as a "random number". Often used is the simple congruential generator:

New_random_number = (Old_random_number * V1 + V2) mod V3

where the constants V1, V2, and V3 are carefully chosen to emulate randomness and keep the series of numbers as large as possible before the same number repeats. Thus, these numbers are not random at all but have a definite sequence to them. [Extra Credit Brain Teaser: Given three consecutive random numbers from a simple congruential generator, how can V1, V2, and V3 be calculated?]

Some computers use the video refresh frame number as a Old_random_number to obtain varied sequences. I once worked on a

program that used this method only to keep getting the same random sequence. It turned out that at the command RUN, the video refresh frame number was reset to zero and the same Old_random_number was always obtained. Another case I read about used an analog to digital converter hooked up to a Geiger counter reading background random radiation to obtain a "true" random number. The only trouble was that the random numbers had a distinct period in them that coincided with a nearby revolving radar.

In any case, stochastic indexing makes use of these reproducible sequences to rapidly regenerate specific information about each indexed feature such as location, size, etc.

## The Demo Program

With regard to the demo program itself, even though it is written in SuperBasic, programmers (FORTRAN, BASIC, etc.) will understand most lines. But a few sections need some additional explanation.

The most important part is the procedure Ith_Cloud. It is used to initially define each cloud's parameters when called from line 230 and is also used to retrieve the ith cloud's parameters when called from line 340.

## Ith Cloud Spacer

In Ith_Cloud, line 470 sets the random number seed for the ith cloud. The 99 times the index is a spacer so that the starting number for each cloud is 99 items further along in the random number sequence. Without the 99, the y location of the first cloud would be the x location of the second cloud (sometimes!). A spacer larger than the largest possible number of parameters for each feature keeps the same sequences from being reused. Sometimes too small a spacer causes no noticeable effect; other times, some weird things happen.

## Acceptance/Rejection

The algorithm as stated so far will give a uniformly random distribution of clouds. That is, any point is just as likely to have a cloud as any other. There are many simulation algorithms to convert uniform distributions to some desired pattern. The one used here is an acceptance/rejection method that allows stochastic indexing to produce varied patterns.

Lines 480 to 510 contain an acceptance/rejection algorithm which is pretty standard in simulation. It works like this: A function is defined over the area that is close to 1 where you want most clouds, close to zero where you want few clouds, and negative where you want no clouds. The function chosen (feel free to chose your own) here is COS(2*PI*Nrows*(x-y)) where Nrows

determines the number of cloud rows and (x-y) locates them in lower-left to upper-right orientation. The Cosine function varies from 1 to -1 as x-y varies.

Next a potential site, x and y, is selected. The function at x and y (-1 to 1 in this case) is compared to a random number (0 to 1). If the function is larger then the random number, the site is selected; but if the random number is larger, then the site is rejected and a new pair of x and y is selected. This continues (REPeat accept_reject) until a site is selected. Do you see why sites with a function near one will have more clouds?

Any function can be used, but make sure it is positive at least in some areas or the REPeat loop will never be exited. Also functions with small areas of positive values and areas with most values near zero will run very long since most sites will be rejected most of the time.

## Speed, Networking, and Portability

Random number generators are generally coded to be very fast. In many cases, a random number can be generated as fast or faster than a number can be fetched from an array in RAM, particularly a multi-dimensional array. If the array must be sent over a LAN (Local Area Network) or WAN (Wide Area Network) for distributed simulation, then stochastic indexing is much faster.

On the other hand, random number generators are often specific to a particular computer. If all computers on the net

are the same, then there is no problem. But if they are different, the random number sequences can be different. There are such things as portable random number generators, but as they say, that is another story.

## REFERENCE

Boehm, A., 1994: "Visual Translucent Algorithm, (VISTA)" **Simulation**, **62**, No.2, 91-97.

LISTING 1

```
100 REMark SID Stochastic Indexing Demo by A.Boehm 30 Aug 1993
110 REMark  *******   Set limits. Try varying these.   *******
120 Seed=898         :REMark defines a new random set. Try varying
it.
130 Total_clouds=100  :REMark Number of clouds.
140 max_puffs=30     :REMark Max Number of puffs for each cloud
150 max_i_size=10   :REMark max size of cluster, closeness of puffs
160 max_puff_size=2 :REMark max size of puff in cluster
170 Nrows=4           :REMark Number of rows per unit (height of
window) distance
180 :
190 WINDOW 512,246,0,0:WINDOW#2,512,10,0,246:PAPER#2,0:INK#2,5
200 REMark  ************  make cloud scape  ************
210 MODE 8:PAPER 1:INK 7:CLS
220 FOR i=1 TO Total_clouds
230    Ith_Cloud i:REMark sets x,y,isize,npuffs_at_i
240    FOR j=1 TO npuffs_at_i
250        CIRCLE x+RND*isize,y+RND*isize,RND*max_puff_size
260    END FOR j
270 END FOR i
280 REMark    ******* find cloud i, puff j ********
290 REPeat find
300          CLS#2:PRINT#2,  'Find  which  cloud?  Enter  1  to
';Total_clouds,:INPUT#2, i
310    IF i<1 OR i>Total_clouds THEN BEEP 2000,100:GO TO 300
320          PRINT#2,  'Cloud  ';i,'Which  puff?  Enter  1  to
';npuffs_at_i;:INPUT#2, j
330    IF j<1 OR j>npuffs_at_i THEN BEEP 2000,100:GO TO 320
340    Ith_Cloud i:REMark sets x,y,isize,npuffs_at_i
350    FOR k=1 TO j-1:a=RND:b=RND:c=RND
360    x= x+RND*isize:y=y+RND*isize:r=RND*max_puff_size
370    PRINT #2,'Push s to stop or another key to go on.';
380    REPeat blink
390       INK 0:CIRCLE x,y,r
400       INK 7:CIRCLE x,y,r
410       a$=INKEY$:IF a$<>'' THEN EXIT blink
420    END REPeat blink
430    IF a$=='S' THEN STOP
440 END REPeat find
450 :
460 DEFine PROCedure Ith_Cloud (i)
470    RANDOMISE i*99+Seed:REMark sets new seed as a function of i
480    REPeat accept_reject
490        x=RND:y=RND
500       d=COS(2*PI*Nrows*(x-y)):IF d>RND THEN EXIT accept_reject
510    END REPeat accept_reject
520    x=x*162:y=y*100:REMark scale to fill screen
530    isize=RND*max_i_size
540    npuffs_at_i=RND(5 TO max_puffs)
550 END DEFine Ith_Cloud
```